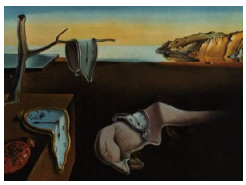

Journal of Statistical Software
MMMMMM YYYY, Volume VV, Issue II. doi: 10.18637/jss.v000.i00

Bayesian Linear Mixed Model with Polygenic Effects

Jing Hua Zhao
University of Cambridge

Jian'an Luan
University of Cambridge

Peter Congdon
University of London

Abstract

We considered Bayesian estimation of polygenic effects, in particular heritability in relation to a class of linear mixed model implemented in R (R Core Team 2016). Our approach is applicable to both family-based and population-based studies in human genetics with which a genetic relationship matrix can be derived either from family structure or genome-wide data. Using a simulated and a real data, we demonstrate our implementation of the models in the generic statistical software systems **JAGS** (Plummer 2013) and **Stan** (**Stan** Development Team 2016c) as well as several R packages. In doing so, we have not only provided facilities in R linking standalone programs such as **GCTA** (Yang *et al.* 2011) and other packages in R but also addressed some technical issues in the analysis. Our experience with a host of general and special software systems will facilitate investigation into more complex models for both human and nonhuman genetics.

Keywords: Bayesian linear mixed models, heritability, polygenic effects, relationship matrix, family-based design, genomewide association study.

1. Introduction

The genetic basis of quantitative phenotypes has been a long-standing research problem associated with a large and growing literature, and one of the earliest was by Fisher (1918) on additive effects of genetic variants (the polygenic effects). In human genetics it is common to estimate heritability, the proportion of polygenic variance to the total phenotypic variance, through twin and family studies. For twin studies, polygenic effects are embedded into correlations between monzygotic and dizygotic twin pairs using the assumption that monozygotic twins share all the genetic materials but half for dizygotic twins. For family studies, the polygenic component is coupled with a relationship matrix in a mixed model with covariates as fixed effects, e.g., Morton and MacLean (1974); Lange (2002). The models differ from those usually seen in general statistics as the polygenic effects are represented by a random variable

that is correlated among all relatives due to genes shared identity-by-descent. The estimation can be inaccurate due especially to shared environment in both twin and family studies.

More recently, a large quantity of single nucleotide polymorphisms (SNPs), single base-pair variants of DNA, available from population-based samples has offered renewed interest in the problem. This is because the data allows for genomic relationship matrix (GRM) to be built as part of a genomewide association study (GWAS) for identification and characterization of the DNA variants and phenotype (our outcome of interest) association. Yang *et al.* (2010) showed that GRM can be used in the mixed model very much the same as in models for families where the relationship matrix is built on familial relationships. Consequently, the ubiquitous availability of DNA also makes the models appropriate for any samples with typed DNA polymorphisms. The approach is applicable to a wide variety of traits including continuous, discrete and time-to-event outcomes (Zhao and Luan 2012). The estimation of heritability (h^2), the proportion of total additive genetic variance as a proportion of total phenotypic variance, is fundamentally important since it largely quantifies the scope of a GWAS in gene discoveries and characterizations.

Bayesian methods are attractive since generic software systems are available to facilitate the model-building, and they also help to address the issue concerning the uncertainty in parameter estimation. Moreover, they give credible intervals with highest probability density (HPD) as opposed to frequentist interval estimates, often derived under simplifying assumptions. Markov chain Monte Carlo (MCMC) serves a practical tool for Bayesian inference with a full characterization of the posterior distribution of the variance components as well as heritability. For this reason, they have been widely used in plant and animal science literature for a broad range of traits, e.g., Yi and Xu (2000); Varona *et al.* (2005). These software almost exclusively use family structure, given that the inverse of the relationship matrix is easily calculated, as was also the case with work on humans, e.g., Burton *et al.* (2005). Exceptions include **BLR** (Perez *et al.* 2010; de los Campos *et al.* 2013) in R (R Core Team 2016) which can accommodate GRM but the analysis often has to be stopped due to nonpositive definite GRM. It is not obvious how these issues can be addressed.

In our own analysis, we have have encountered various issues. Our attempts to tackle of these problems have led to some useful results, which we believe will facilitate similar analyses by other colleagues. Via a simulated data and a real data, we implemented the models using **JAGS** (Just Another Gibbs Sampler) (Plummer 2013), **Stan** (Sampling Through Adaptive Neighborhoods) (**Stan** Development Team 2016c) and in the case of large sample **BLR**. We wrote utilities in R to read or write GRM as generated from software **GCTA** (Yang *et al.* 2011) to be used in these software, which contain functions to calculate heritability and its standard error when polygenic and residual variance/standard errors are given. We further adapted **MCMCglmm** (Hadfield 2010) to enable comparison between family-based or genotype-based relationship matrices. These functions are available from the R package **gap** (Zhao 2007) with further information. We also gave expression for perturbing the covariance matrix when GRM is considered nonpositive definite. We believe our work will be of interest in human genetics as well as animal and plant genetics. Below we will briefly describe the polygenic model, a simulated data as a benchmark and an application. We then conclude with a summary, which includes generic discussions on non-genetic effects, missing outcomes, efficient implementation, frequentist and Bayesian estimates of heritability for the **GCTA** documentation example.

2. Statistical models

We start with an outline of the linear mixed model, showing how total additive genetic effects can be framed with respect to a relationship matrix. We then consider specification of the Bayesian linear mixed model.

2.1. Linear mixed model

To motivate we consider a study of body mass index (BMI, body weight/height (kg/m^2)) in relation to sex (0 = Man, 1 = Woman) and age (in years). A linear model (LM) of BMI on sex and age is as follows,

$$\text{BMI} \quad = b_0 + b_1 \text{ sex} + b_2 \text{ age} + e \tag{1}$$

where b_0 is an intercept, b_1 and b_2 are the regression coefficients for sex and age, indicating a unit change in BMI attributable to being a woman than man and per-year increase in age, respectively. e is a residual term indicating effects on BMI other than sex and age. As will soon become clear, there is a need to have extra terms which are random variable, leading to a linear mixed model (LMM). More generally, let y be a continuous variable and our outcome of interest, X covariates, u random effects, a LMM has the following form,

$$y \quad = \quad X\beta + Zu + e \tag{2}$$

where

y – an $N \times 1$ vector of observations

X – an $N \times p$ matrix of known covariates

β – a $p \times 1$ vector of unknown regression coefficients

Z – a known $N \times q$ matrix of variables

u – a $q \times 1$ vector of unknown random effects

e – an $N \times 1$ vector of (unobservable random) errors

We assume that $u \sim N(0, D)$ and $e \sim N(0, E)$, so that $y \sim N(X\beta, V)$ with $V = E + ZDZ^\top$. Statistical inference of this model, based on the frequentist approach, can be done with maximum likelihood (ML) or restricted maximum likelihood (REML). whose procedures are widely available (see Sorensen and Gianola 2002, for further details).

2.2. Linear mixed model with polygenic effects

We assume that our trait of interest, y, is a function of m causal variants each with effect u_i, $u_i \sim N(0, \sigma_u^2)$, $i = 1, \ldots, m$, treated as random effect, σ_u^2 a polygenic variance. These variants are DNA polymorphisms at particular positions across the genome. At locus i, we assume the two causal alleles are q and Q with frequency $1 - f_i$, f_i, and forms genotypes qq, qQ and QQ, respectively with additive effects 0, 1, and 2. The genotypic effects are associated with Binomial distribution, $Bin(2, f_i)$, with mean $2f_i$ and variance $2(1 - f_i)f_i$, respectively, leading to normalized additive effects (z_i) being $-2f_i/\sqrt{2(1 - f_i)f_i}$, $(1 - 2f_i)/\sqrt{2(1 - f_i)f_i}$

and $(2-2f_i)/\sqrt{2(1-f_i)f_i}$. The simplest form of polygenic model uses a linear combination of effects from all causal variants, i.e., $g = \sum_{i=1}^{m} z_i u_i$ where z_i can be seen as a function of the frequency of allele with effect acting as a scaling factor such that $\mathsf{E}z_i = 0$ and $\mathsf{VAR}(z_i) = 1$. In matrix notation $g = Zu$, we have $g \sim N(0, \sigma_u^2 ZZ^\top)$ and $\sigma_g^2 = m\sigma_u^2$ is the variance of total additive effects ("polygenic effects"). From this $\mathsf{VAR}(y) = \sigma_u^2 ZZ^\top + \sigma^2 I = \sigma_g^2 ZZ^\top/m = \sigma_g^2 A + \sigma^2 I$, where $A = ZZ^\top/m$ amounts to a relationship matrix and indeed called a GRM at the causal loci, σ^2 is the residual variance, and I an identity matrix. Heritability is defined as the proportion of phenotypic variance explained by the polygenic effects, namely, $h^2 = \sigma_g^2/(\sigma_g^2 + \sigma^2)$.

The matrix A can be represented with genomewide data containing a large number (M) of SNPs analogous to causal variants, i.e., $G = WW^\top/M$ where $w_{ij} = (x_{ij} - 2p_i)/\sqrt{2(1-p_i)p_i}$, $j = 1, 2, 3$ represents the genotypic effects of SNP i and p_i is the allele frequency, while $x_{ij} = 0, 1, 2$ for SNP i having alleles a_1, a_2, and genotypes a_1a_1, a_1a_2, a_2a_2, respectively. A series of refinements of the G matrix has been suggested by Yang *et al.* (2010). The **GCTA** software can generate a compressed (.grm.gz) or binary (.grm.bin) form of GRMs from genomewide SNPs and provide REML estimates for the polygenic model.

In summary, our model is similar to (2) in that $D = \sigma_g^2 G$ and $E = \sigma^2 I$, where G is a GRM,

$$y \;=\; X\beta + g + e \tag{3}$$

$\mathsf{VAR}(y) = \sigma_g^2 G + \sigma^2 I$ with g being "polygenic effects" and G an $N \times N$ GRM.

For data on relatives, the additive genetic relationship matrix A can also be derived from given family structure which is twice the kinship matrix (Lange 2002) whose entries represent probabilities of genes shared identity-by-descent among pairs of relatives. The matrix can be generated by a number of R packages such as **kinship2** (Therneau and Sinnwell 2015) at the Comprehensive R Archive Network (CRAN).

2.3. Bayesian linear mixed model with polygenic effects

A Bayesian linear mixed model (BLMM) with polygenic effects follows the set-up above, whose sampling model is as follows,

$$\begin{aligned} y|\beta, u, \sigma^2 &\sim N(X\beta + Zu, \sigma^2 I) \\ \beta|\sigma_\beta^2 &\sim N(0, \sigma_\beta^2 B) \\ u|\sigma^2 &\sim N(0, \sigma^2 A) \end{aligned} \tag{4}$$

where B is a known, nonsingular matrix and σ_β^2 is a hyperparameter. Full specification of the model is furnished with appropriate distributions for the variance components, e.g., Section 6.3 of Sorensen and Gianola (2002). For the polygenic model (3) in this paper, we have likelihood and assumed prior specifications as follows:

$$\begin{aligned} y &\sim N(\mu, \sigma^2 I) \\ \mu &= X\beta + g \\ \beta_j &\sim N(0, 1000^2), j = 1, \ldots, p \\ g &\sim N(0, \sigma_g^2 G) \\ \sigma_g^2 &\sim InvGamma(s_1, s_2) \\ \sigma^2 &\sim InvGamma(s_1, s_2) \end{aligned} \tag{5}$$

where s_1 and s_2 are chosen to provide noninformative priors, and the matrix B is diagonal. Other priors for the variance components such as uniform are possible, as in Section 4.2 below, Waldmann (2009) and Gelman (2006).

2.4. Handling of the G matrix

Simulation of the polygenic effects in section 2.3 involves multivariate Normal distribution, which could be very time-consuming when N gets large. A speedup can be achieved by obtaining the precision matrix as input to software described below. More often, a Cholesky decomposition can be applied. For $g \sim N(0, \sigma_g^2 G)$, Let $G = CC^\top$ and $z_i \sim N(0,1)$, $i = 1, \ldots, N$, then $g_i = \sigma_g C z_i \sim N(0, \sigma_g^2 G)$. As expression (5) is amenable to a few environments for MCMC, these are exposed in Section 3.2 below.

3. Benchmark

Data from Meyer (1989) as in Tempelman and Rosa (2004) is used as our benchmark. The pedigrees for each of these 282 animals derive from an additional 24 base population (Generation 0) animals that do not have records of their own, nevertheless are of interest with respect to the inference on their own additive genetic values. Furthermore, it is presumed that these original 24 base animals are not related to each other. Therefore, the row dimension of u is 306 (282+24). To facilitate discussions the data is made available from **gap** at CRAN.

3.1. Frequentist approach

Tempelman and Rosa (2004) gave a variety of estimates using **SAS** (SAS Institute Inc. 2014). We are interested in the REML estimates which are available from **regress** (Clifford and McCullagh 2006).

```
R> set.seed(1234567)
R> meyer <- within(meyer, {
+    y[is.na(y)] <- rnorm(length(y[is.na(y)]),
+      mean(y, na.rm = TRUE), sd(y, na.rm = TRUE))
+    g1 <- ifelse(generation == 1, 1, 0)
+    g2 <- ifelse(generation == 2, 1, 0)
+    id <- animal
+    animal <- ifelse(!is.na(animal), animal, 0)
+    dam <- ifelse(!is.na(dam), dam, 0)
+    sire <- ifelse(!is.na(sire), sire, 0)
+  })
R> G <- kin.morgan(meyer)$kin.matrix * 2
R> library("regress")
R> r <- regress(y ~ -1 + g1 + g2, ~G, data = meyer)
R> r

Likelihood kernel: K = g1+g2

Maximized log likelihood with kernel K is  -843.962
```

```
Linear Coefficients:
    Estimate Std. Error
 g1  222.994      1.429
 g2  238.558      1.760

Variance Coefficients:
    Estimate Std. Error
 G    31.672     13.777
 In   72.419     10.182
```

```
R> with(r, h2G(sigma, sigma.cov))
```

```
Vp = 104.091 SE = 9.925092
h2G = 0.3042677 SE = 0.1147779
```

Note that we deliberately filled the missing data according to the observed (We will relax this later on), then employed the `kin.morgan` function to obtain the kinship matrix, which is in turn used by the `regress` function from **regress** package. We have $h^2(SE) = 0.30(0.11)$.

3.2. Bayesian approach

We now turn to Bayesian approach and begin with generic implementation in the Bayesian inference Using Gibbs Sampling (BUGS) As most such implementation would involve large sample, we moved away from **WinBUGS** (Lunn *et al.* 2000) and used **OpenBUGS** (**OpenBUGS** Foundation 2015) and **JAGS** under Linux. Both allow for command line execution but as noted earlier (Sturtz *et al.* 2005) data manipulation is required which can be greatly facilitated with **OpenBUGS**, specifically using the R package **R2OpenBUGS** (Sturtz *et al.* 2005). We focused on **JAGS** as it was better tuned under Linux with **LAPACK** (Anderson *et al.* 1999), or **Intel MKL**, (Intel 2013) and the R counterpart **R2jags** (Su and Yajima 2015). We use multiple chains (e.g., 2 to 4), and Brooks-Gelman-Rubin (BGR) statistics, provided in **JAGS** or **Stan**, to check convergence. Initial parameter values are generally based on subject matter knowledge and/or parameter estimates from classical estimation.

JAGS

First, we prepare for the data in R and call **JAGS** via **R2jags**,

```
R> C <- chol(G)
R> N <- dim(meyer)[1]
R> data <- with(meyer,
+    list(N = N, y = y, g1 = g1, g2 = g2, u = rep(0,N), GI = solve(G))
+  )
R> inits <- function()list(b1 = 0, b2 = 0, tau.p = 0.03, tau.r = 0.014)
R> parms <- c("b1", "b2", "p", "r", "h2")
```

We apply inverse gamma priors

```
R> modelfile <- function() {
+    b1 ~ dnorm(0, 0.000001)
+    b2 ~ dnorm(0, 0.000001)
```

```
+    tau.p ~ dgamma(0.001, 0.001)
+    tau.r ~ dgamma(0.001, 0.001)
+    sigma.p <- 1 / sqrt(tau.p)
+    sigma.r <- 1 / sqrt(tau.r)
+    g[1:N] ~ dmnorm(u[], GI[,] / p)
+    for (i in 1:N) {y[i] ~ dnorm(b1 * g1[i] + b2 * g2[i] + g[i], tau.r)}
+    p <- pow(sigma.p, 2)
+    r <- pow(sigma.r, 2)
+    h2 <- p / (p + r)
+  }
R> library("R2jags")
R> jagsfit <- jags(data, inits, parms, modelfile,
+    n.chains = 2, n.burnin = 500, n.iter = 5000)
```

Like **OpenBUGS**, the Normal distribution in **JAGS** is specified with respect to precision. The `solve` function returns the inverse so it is only calculated once. The results are very close to REML estimates.

```
Inference for Bugs model at "/tmp/RtmpNuBbQo/model66b330e4d3e4.txt", fit using jags,
 2 chains, each with 5000 iterations (first 500 discarded), n.thin = 4
 n.sims = 2250 iterations saved
          mu.vect sd.vect     2.5%      25%      50%      75%
b1        222.966   1.466  220.230  221.939  222.923  223.900
b2        238.553   1.824  235.063  237.345  238.555  239.716
h2          0.296   0.085    0.151    0.234    0.287    0.349
p          31.368  10.660   14.861   23.501   29.772   37.511
r          73.659   8.807   57.105   67.752   73.315   79.419
deviance 2181.707  26.524 2123.499 2165.236 2183.819 2200.407
           97.5%  Rhat n.eff
b1        226.054 1.001  2200
b2        242.257 1.003  2200
h2          0.476 1.040    52
p          56.229 1.037    53
r          91.978 1.010   160
deviance 2227.942 1.020    81

For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=1).

DIC info (using the rule, pD = var(deviance)/2)
pD = 347.5 and DIC = 2529.2
DIC is an estimate of expected predictive error (lower deviance is better).
```

The version with Cholesky decomposition is as follows, noting that the factored matrix needs to be transposed. We also use uniform priors.

```
R> data <- with(meyer,list(N = N, y = y, g1 = g1, g2 = g2, C = t(C)))
R> inits <- function() list(b1 = 0, b2 = 0, sigma.p = 0.03, sigma.r = 0.014)
```

```
R> modelfile=function() {
+    b1 ~ dnorm(0, 0.001)
+    b2 ~ dnorm(0, 0.001)
+    sigma.p ~ dunif(0, 1000)
+    sigma.r ~ dunif(0, 1000)
+    p <- pow(sigma.p, 2)
+    r <- pow(sigma.r, 2)
+    tau <- pow(sigma.r, -2)
+    g[1:N] <- sigma.p * C[,] %*% z[]
+    for (i in 1:N) {z[i] ~ dnorm(0, 1)}
+    for(i in 1:N) {y[i] ~ dnorm(b1 * g1[i] + b2 * g2[i] + g[i], tau)}
+    h2 <- p / (p + r)
+  }
R> jagsfit2 <- jags(data, inits, parms, modelfile,
+    n.chains = 2, n.burnin = 500, n.iter = 5000)
```

where we also used uniform priors for the variance components, and the results are similar.

```
Inference for Bugs model at "/tmp/RtmpNuBbQo/model66b326b1ea89.txt", fit using jags,
 2 chains, each with 5000 iterations (first 500 discarded), n.thin = 4
 n.sims = 2250 iterations saved
          mu.vect sd.vect     2.5%      25%      50%      75%
b1        222.190   1.434  219.321  221.209  222.226  223.122
b2        237.476   1.744  234.113  236.300  237.504  238.602
h2          0.300   0.083    0.150    0.240    0.295    0.357
p          32.163  10.609   15.147   24.434   30.707   38.622
r          74.108   8.738   58.018   68.189   73.666   79.857
deviance 2181.679  26.174 2124.323 2165.462 2183.597 2200.069
            97.5%  Rhat n.eff
b1        224.983 1.003  1900
b2        240.773 1.002  2200
h2          0.471 1.001  1800
p          55.611 1.001  2200
r          92.640 1.001  2200
deviance 2226.742 1.001  2200

For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=1).

DIC info (using the rule, pD = var(deviance)/2)
pD = 342.7 and DIC = 2524.3
DIC is an estimate of expected predictive error (lower deviance is better).
```

Stan

We further experimented with **Stan**, which is appealing to us as it implemented faster sampling algorithms (Gelman *et al.* 2014, p. 307). We worked on both the R interface, **rstan** (**Stan** Development Team 2016b), and command line version, **cmdstan** (**Stan** Development Team 2016a).

```
R> data <- with(meyer,list(N = N, y = y, g1 = g1, g2 = g2, G = G))
R> library("rstan")
R> meyer.stan = '
+  data {
+    int N;
+    vector[N] y;
+    vector[N] g1;
+    vector[N] g2;
+    matrix[N, N] G;
+  }
+  transformed data {
+    matrix[N,N] C;
+    C = cholesky_decompose(G);
+  }
+  parameters {
+    vector[2] b;
+    vector[N] z;
+    real sigma_p2;
+    real sigma_r2;
+  }
+  transformed parameters {
+    real sigma_p;
+    real sigma_r;
+    vector[N] g;
+    sigma_p = sqrt(sigma_p2);
+    sigma_r = sqrt(sigma_r2);
+    g = sigma_p * C * z;
+  }
+  model {
+    b ~ normal(0, 1000);
+    sigma_p2 ~ inv_gamma(0.001, 0.001);
+    sigma_r2 ~ inv_gamma(0.001, 0.001);
+    z ~ normal(0, 1);
+    y ~ normal(b[1] * g1 + b[2] * g2 + g, sigma_r);
+  }
+  generated quantities {
+    real h2;
+    real p;
+    real r;
+    p = sigma_p2;
+    r = sigma_r2;
+    h2 = p / (p + r);
+  }
+  '
R> parms = c("b", "p", "r", "h2")
R> f1 = stan(model_code = meyer.stan, data = data, chains = 2, iter = 500,
+    verbose = FALSE)
```

```
R> f2 = stan(fit = f1, data = data, chains = 2, iter = 5000, pars = parms,
+    verbose = FALSE)
```

where results from the first `stan` call is given to the second formal call. Note that the program is sectioned with `data` passed from R and part of which is in `transformed data`. These are followed by `parameters` and `transformed parameters` before they are used in `model`. Our quantities of interest can further be obtained from `generated quantities`.

The results from **Stan** are shown below and Figure 1,

```
Inference for Stan model: df0c4ce12df598b4fcdd553dfe7d2cee.
2 chains, each with iter=5000; warmup=2500; thin=1;
post-warmup draws per chain=2500, total post-warmup draws=5000.

        mean se_mean     sd    2.5%     25%     50%     75%   97.5%
b[1] 222.960   0.032  1.452 220.230 221.979 222.928 223.900 225.839
b[2] 238.566   0.034  1.726 235.228 237.414 238.560 239.703 242.080
p     31.038   0.332 10.413  14.156  23.512  29.734  37.403  55.031
r     73.904   0.218  8.743  57.302  68.048  73.694  79.606  91.807
h2     0.293   0.003  0.084   0.145   0.233   0.288   0.347   0.470
     n_eff  Rhat
b[1]  2066 1.000
b[2]  2573 1.000
p      985 1.005
r     1613 1.002
h2     934 1.005

Samples were drawn using NUTS(diag_e) at Sun Mar  4 22:27:07 2018.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).

Potential scale reduction factors:

     Point est. Upper C.I.
b[1]          1          1
b[2]          1          1
p             1          1
r             1          1
h2            1          1
lp__          1          1

Multivariate psrf

1
```

where the BGR diagnostic statistics show convergence of the parameters. The overlapped density plots for the two chains are also shown,

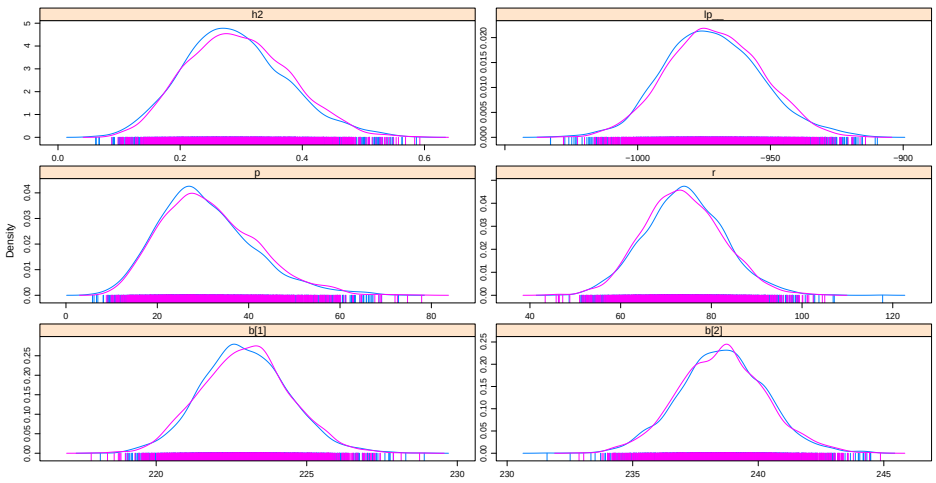

Figure 1: Density plot for the Meyer data from **Stan**.

Although both **OpenBUGS** and **JAGS** work as standalone programs, the counterpart in **Stan**, **cmdstan**, is much easier. We simply need to make a copy of the program above, say `meyer.stan`, to the cmdstan directory and issue "stanc" to generate the C++ source or even "make meyer" to generate the executable, We first prepare for our data in R and then use functions `bugs.data` and `bugs2jags` to output an input file for `meyer`,

```
R> library("R2OpenBUGS")
R> data <- with(meyer, list(N = N, y = y, g1 = g1, g2 = g2, G = G))
R> bugs.data(data, data.file = "meyer_bugs.txt")
[1] "meyer_bugs.txt"
R> library("coda")
R> bugs2jags("meyer_bugs.txt", "meyer_stan.txt")
```

and we can call

```
./meyer sample data file=meyer_stan.txt output file=meyer.csv
stansummary meyer.csv
```

The data file (meyer_stan.txt) is used by the executable to generate our output in `meyer.csv`, and the summary statistics are given by the `print` utility. Equally, **rstan** can also pick up

results to allow for graphical facilities in R.

4. Additional considerations

4.1. Parallel computation

It is possible to take advantage of multicore facility in **R** for multiple chains via package **parallel** (R Core Team 2016), back to the Meyer data it can be done as follows,

JAGS

```
attach(meyer)
library("R2jags")
out <- jags.parallel(data, inits, parms, modelfile,
                     n.chains = 4, n.burnin = 500, n.iter = 5000)
detach(meyer)
```

Somehow the data needs to be attached.

Stan

```
library("parallel")
parms <- c("b","p","r","h2")
f1 <- stan(model_code=meyer.stan, data = data, chains = 4, iter = 500,
  verbose = FALSE)
l <- mclapply(1:4, mc.cores = 4, function(i)
  stan(fit = f1, seed = 12345, data = data, iter=5000,
  chains = 1, chain_id = i, refresh = -1))
f2 <- sflist2stanfit(l)
```

One can use `detectCores()` function to obtain the number of cores on the system and here four chains are run in parallel. Alternatively, a call can be made with

```
options(mc.cores = parallel::detectCores() - 1)
```

4.2. Nonpositive definite G matrix

We found it more likely to have a nonpositive definite G matrix in (4, 5) than a kinship matrix. In theory, we can get around this with a perturbation (ϵ) as described in (Guo and Thompson 1991, p. 174), namely to replace G with $\tilde{G} \equiv (G + \epsilon/\sigma_g^2 I)$, so that $\sigma_g^2 \tilde{G} = \sigma_g^2 G + \epsilon$ and $\tilde{\sigma}^2 = \sigma^2 - \epsilon$ one only needs to amend σ^2 as $\tilde{\sigma}^2 + \epsilon$. The is according to the Gerschgorin theorem (Varga 2004, theorem 1.4) as popularized by ridge regression.

```
modelfile <- function() {
    b1 ~ dnorm(0, 0.000001)
    b2 ~ dnorm(0, 0.000001)
    sigma.p ~ dunif(0, 1000)
    sigma.r ~ dunif(0, 1000)
    p <- pow(sigma.p, 2)
    r <- pow(sigma.r, 2)
```

```
    tau <- pow(sigma.r, -2)
    g[1:N] ~ dmnorm(u[], inverse(p * G[,] + eps * I[,]))
    for(i in 1:N) {y[i] ~ dnorm(b1 * g1[i] + b2 * g2[i] + g[i], tau)}
    h2 <- p / (p + r)
}
```

This will be the same as before when $\epsilon = 0$. While this is mathematically viable, it involves additional matrix inversion in **JAGS** making our task even more formidable for MCMC convergence. We used $(G + \epsilon I)$ in place of the relationship matrix and $\sigma^2 + \epsilon\sigma_g^2$ as residual variance, which do not involve direct simulation from multivariate Normal distribution.

5. Application: familial vs genomic heritabilities

The data used in this section was derived from a large family study which mirrors work by (Klimentidis *et al.* 2013), to enable contrasting genetic relationship from family structure and genome-wide data.

5.1. Frequentist approach

Two relationship matrices based on family structure and genomic data were generated by R and **GCTA**, respectively, to be used by **GCTA** for REML estimation.

The genetic relationship matrix was built from pedigree structures with **kinship** (Atkinson and Therneau 2012),

```
trios <- read.table("trios.dat",header=TRUE)
library("kinship")
kmat <- with(trios, kinship(id, fid, mid))
id <- trios[c("pid", "id")]
N <- dim(trios)[1]
M <- rep(N, N * (N + 1) / 2)
library("gap")
WriteGRM("PRM", id, M,  2 * kmat)
```

which was used by **GCTA** for REML estimates. Assuming that phenotype and covariate information are stored in bmi.dat and covars.dat, **GCTA** can be called as follows,

```
gcta64 --reml --grm-gz PRM --pheno p.dat --out PRM --thread-num 10
```

The REML estimates were obtained with **GCTA** as follows,

```
gcta64 --reml --grm-gz GRM --pheno p.dat --out GRM --thread-num 10
```

Note the calls to **GCTA** should be run under the Linux shell directly. The results are shown in Table 5.1, where l_0 and l are the log-likelihoods with and without the polygenic component, respectively. **GCTA** gave estimates of heritability which was remarkably similar, where $h^2(SE)$ equals 0.46 (0.02) and 0.47 (0.03), respectively for genome-based and family-based estimates. One may rather use genomic structure as it is associated with a greater likelihood.

5.2. Bayesian approach

Besides results from REML shown above, in a separation analysis on lung function from the same cohort, the two approaches yielded almost identical heritability estimates (Klimentidis

	Genomic data		Family structure	
	Variance components	SE	Variance components	SE
σ_g^2	10.38	0.64	10.62	0.74
σ^2	12.33	0.50	12.01	0.63
$h^2 = \sigma_g^2/(\sigma_g^2 + \sigma^2)$	0.46	0.02	0.47	0.03
l_0	-13479.85		-13572.17	
l	-13724.35		-13724.35	
$\chi^2 = -2(l - l_0)$	489.00		304.37	

Table 1: Estimates based on familial and genomic relationship matrices.

et al. 2013). The marked difference in deviance prompted us to seek to characterize variability of heritability in a Bayesian framework.

For this "large N" ($N \gg 1,000$) problem, the implementation in either **JAGS** or **Stan** became prohibitively slow, we therefore resorted to specific implementations **MCMCglmm** and **BLR** that we were aware of. However, an adaption of **MCMCglmm** with GRM took about three days on our Linux system with 300 burn-ins and 1,000 iterations and it is infeasible to consider large number of iterations. As with **BLR**, we encountered the issue of nonpositive definite GRM. While adding a perturbation to the GRM it was not clear how our results will be adjusted. We also sought for the possibility of approximate Bayesian methods through which **AnimalINLA** (Holand *et al.* 2013) came to our attention. It was derived from **INLA** (Integrated Nested Laplace Approximation) (Rue *et al.* 2014). It was not obvious it can handle GRM but we would like to explore.

First, we set up the data to be used,

```
pheno <- read.table("p.dat",col.names=c("pid","id","r"))
N <- nrow(pheno)
trios[trios==0] <- NA
f <- merge(pheno, trios[,-1], by = "id", all = TRUE)
p <- data.frame(f[with(f,order(pid,id)),], u = 1:N, e = 1:N)
rownames(p) <- 1:N
```

AnimalINLA

The **AnimalINLA** package was used first taking family structures.

Using family structure

```
R> library("AnimalINLA")
R> library(pedigree)
R> trios <- add.Inds(p[c("id","fid","mid")])
R> trios[is.na(trios)] <- 0
R> data <- merge(trios,p[c("id","r")],by="id",all.x=TRUE)
R> nr <- nrow(data)
R> p2 <- data.frame(data,u=1:nr,e=1:nr)
R> p2 <- within(p2,id <- as.integer(id))
R> xx <- compute.Ainverse(p2[c("id", "fid", "mid")])
```

```
R> fit <- animal.inla(
+    response = "r", fixed = NULL,
+    genetic = "id", Ainverse = xx,
+    type.data = "gaussian", data = p2,
+    sigma.e = TRUE, dic = TRUE)
R> save(p2, xx, fit, file = "AnimalINLA.fit")
```

The computation was done in minutes on our Linux with the default setup and the output is as follows,

```
R> with(fit,summary.hyperparam)
                      mean         sd 0.025quant   0.5quant 0.975quant
Heritability     0.4630103 0.02598916  0.4120106  0.4630394   0.514006
Variance for id 10.5193350 0.56499203  9.2356416 10.4878660  11.885420
Variance for e  12.1880055 0.46143554 11.1449546 12.1680929  13.307111
```

The R S3 function `plot.Animalinla` always sets `xlim=c(0, 1)` and created plots on the console so we revised this.

```
R> par(mfrow=c(3,1))
R> plot.default(sigma.u, type = "l", ylab = "Posterior",
+    xlab = expression(paste(sigma[u]^2)))
R> plot.default(sigma.e, type = "l", ylab = "Posterior",
+    xlab = expression(paste(sigma^2)))
R> plot.default(gaussian.h, type = "l", ylab = "Posterior",
+    xlab = expression(paste(h^2)), xlim = c(0, 1))
```

The posterior distribution of h^2 is shown in Figure 2.

Using GRM

By inspecting structure of object `xx` above, we avoid `compute.Ainverse` function and constructed a compatible object as follows,

```
g <- ReadGRM("GRM")
gi <- solve(g$GRM)
k2l <- matrix(NA, N * (N + 1) / 2, 3)
L <- 1
for (i in 1:N) {
    for (j in 1:i) {
       k2l[L, ] <- c(j, i, gi[i, j])
        L <- L + 1
    }
}
x <- list()
x$Ainverse <- k2l[k2l[, 3] != 0, ]
x$map <- cbind(trios[, 2], 1:N)
class(x) <- "ped"
fit <- animal.inla(response = "r",
                   fixed = NULL, genetic = c("id"),
                   Ainverse = x, type.data = "gaussian",
                   data = bmi, sigma.e = TRUE, dic = TRUE)
```

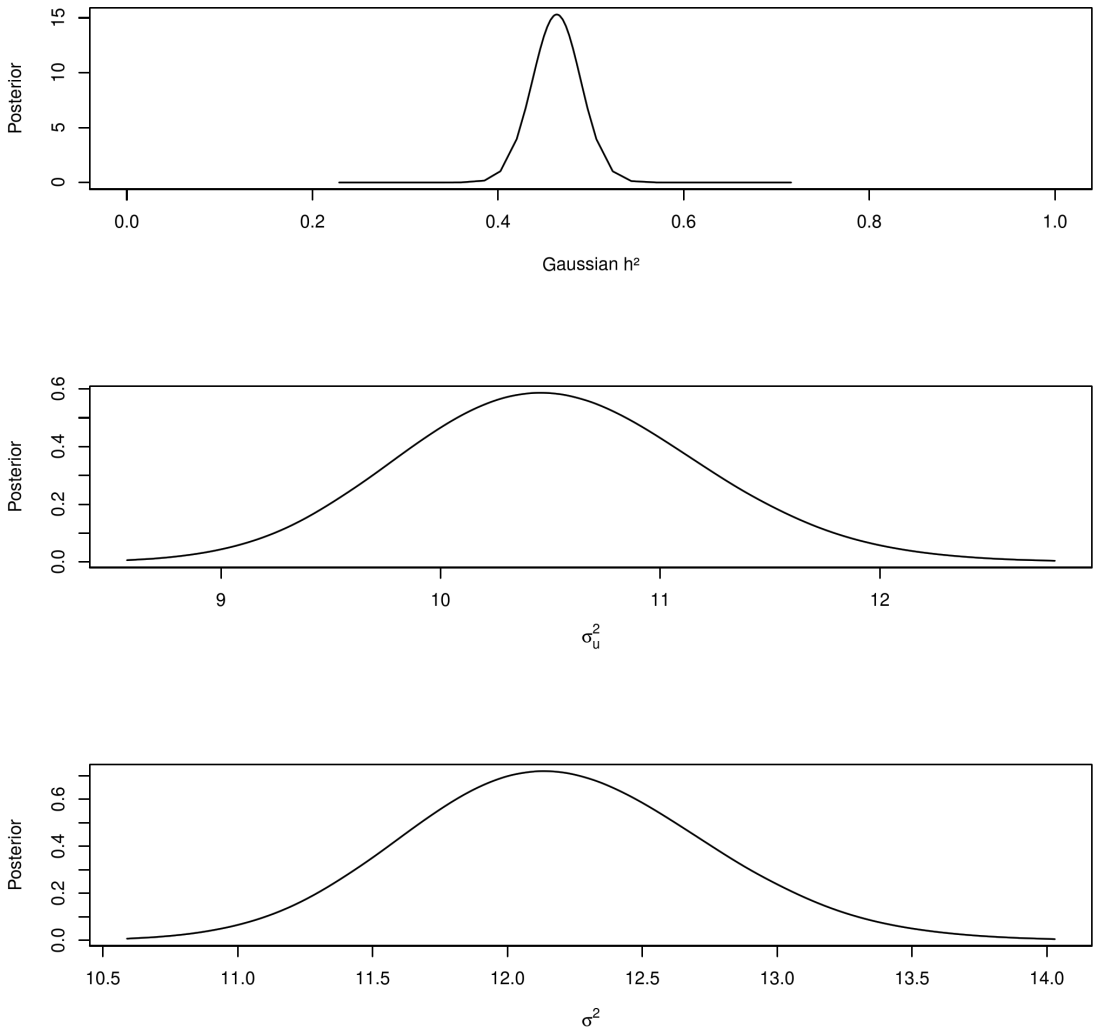

Figure 2: Posterior distributions according to **AnimalINLA**.

where function `ReadGRMBin` reads in the `GRM.grm.gz`, and `GRM.grm.id` as generated from **GCTA** in Section 5.1 into object `g`, which is in turn inversed and transferred into a long-format matrix called `k2l` and fed into `animal.inla`. Unfortunately compared to the version using family structure, the running time for this implementation is prohibitively long on our Linux system.

BLR

Our call is as follows,

```
R> y <- as.matrix(r)
R> eps <- 0.1
R> m <- BLR(y,
+    GF = list(ID = 1:N, A = g$GRM+diag(eps, N)),
+    prior = list(varU = list(df = 3, S = 4),
+    varE = list(df = 3, S = 4)),
+    nIter = 300000, burnIn = 150000, thin = 1, saveAt = "fgh.BLR_")
R> attach(m)
R> varU
          [,1]
[1,] 11.59946
R> varE + varU * eps
          [,1]
[1,] 12.11292
R> varU / ((1 + eps) * varU + varE)
           [,1]
[1,] 0.4891733
R> detach(m)
R> U <- scan("fgh.BLR_varU.dat")
R> E <- scan("fgh.BLR_varE.dat") + eps * U
R> e <- as.mcmc(cbind(U, E, h2 = U / (U + E)))
R> summary(e)$statistics
R> HPDinterval(e, probs = 0.95)
      Mean      SD  Naive SE Time-series SE
U  10.3582 0.64494 0.0010902      0.0067283
E  11.3098 0.55301 0.0009348      0.0054231
h2  0.4561 0.02396 0.0000405      0.0002591

        lower     upper
U   9.1166080 11.647820
E  10.2278200 12.397340
h2  0.4092243  0.503218
attr(,"Probability")
[1] 0.95
```

the columns and rows of the GRM are indexed in the object `g$id` whose ordering was used to compromise with that of the phenotypic data. The argument `bF` specifies flat priors for

regression coefficients earlier. The argument `A` is "symmetric, positive definite" matrix (de los Campos *et al.* 2013). The priors for the polygenic (`varU`) and residual (`varE`) variances follow de los Campos *et al.* (2013) as scaled inverse χ^2 with expectation $S/(df-2), S = var(y)(1-h^2)(df-2)$. This is roughly the same for both variances. The perturbation $\epsilon = 0.1$ has enabled the GRM to be positive definite. Note that the `saveAt` option informs the function to keep values of `bF`, `varU` and `varE` at each iteration to `fgh.BLRbF.dat`, `fgh.BLRvarU.dat` and `fgh.BLRvarE.dat`, respectively. Figure 3 shows the results of a very long chain (150,000 burn-ins, 300,000 iterations). The sequences were also converted into an mcmc object of **coda** from which we obtained the `HPD` interval via function `HPDinterval`. The density plot is indeed similar to Figure 2.

```
R> plot(e)
```

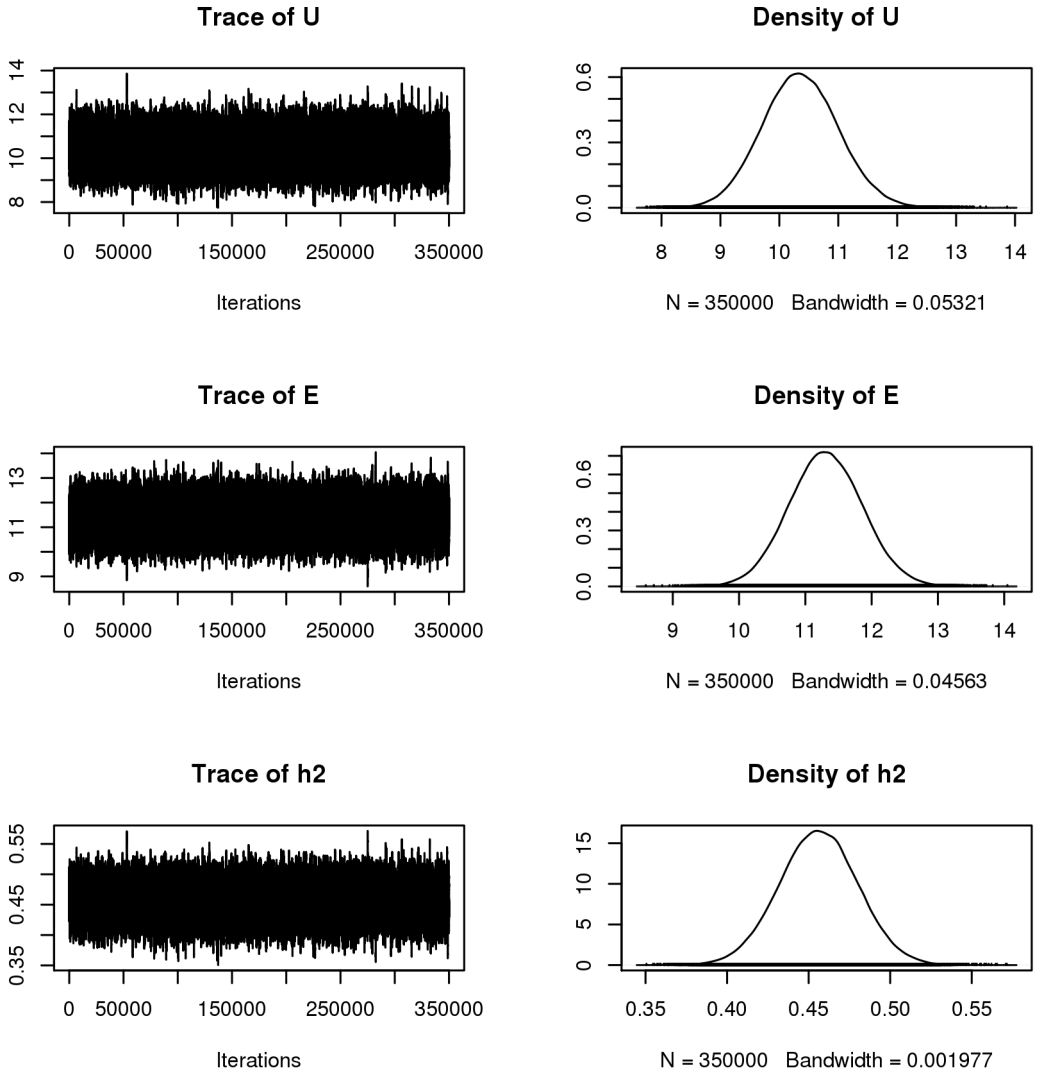

Figure 3: Posterior distributions of polygenic variance (top), residual variance (middle) and h^2 (bottom) according to **BLR**.

6. Summary

We implemented Bayesian linear mixed models that involve a direct use of the relationship matrix. Generic software such as **JAGS** or **Stan** renders greater simplicity than purpose-written software and more flexibility for complex models. Through data analysis we showed that the frequentist and Bayesian approaches can give comparable point estimates but the latter is desirable with its ability to use prior information and produce posterior distributions. For large samples, unlike the usual availability of family structures and therefore fast on-the-fly calculation of the inverse of the precision matrix involving polygenic variance (Waldmann 2009; Damgaard 2007) they have great difficulty in dealing with large genomic matrice(s). We therefore exploited matrix decomposition and parallel computation. We also compiled **JAGS** using both **LAPACK** and **Intel MKL**. Given that the computing time remains prohibitive, we further used approximate Bayesian inference such as Laplace approximation, in particular **INLA** as in **AnimalINLA**, which was again humbled by the high dimensionality and non-sparsity density of the GRM. Our analysis also naturally called up a number of packages in the R system with its ability for data management, powerful programming and modeling.

The implementation has not been seen in the literature and **Stan** gave comparable results to the usual **REML** and **JAGS**. Our setup enables relationship matrix from either family or population data directly into a polygenic model. The comparison of both types of relationship matrices is now possible with **MCMCglmm** from which a function `MCMCgrm` was implemented in **gap**. **BLR** runs faster but would fail with a non-positive definite G matrix. Unlike Guo and Thompson (1991), our approach does not involve repeated inversion or factorization of the variance-covariance matrix at the sampling stage and has enabled analysis with **BLR**. The analysis also went beyond our previous experiment (Zhao and Luan 2012), whose focus was only on frequentist approaches. A reviewer pointed to us work by Bae *et al.* (2014) noting previous work on on decomposition and conditioning by (Waldmann *et al.* 2008; Hallander *et al.* 2010) which have "proposed an approach based on a decomposition of the multivariate normal distribution of the random effects into univariate normal distributions using conditional distributions" but "fails to produce accurate results with large multigenerational families" though the authors "were not able to pinpoint the reason for the apparent discrepancy" (between the conditioning and singular value decomposition). In essence, the model as in Bae *et al.* (2014) has a covariance structure

$$V = 2\sigma_g^2 \begin{pmatrix} K_1 & & & \\ & K_2 & & \\ & & \ddots & \\ & & & K_m \end{pmatrix} + \sigma^2 I \tag{6}$$

where K_i are the kinship matrices associate with a particular family i, $i = 1, \ldots, m$. In our case, the GRM does not have the block structure. In Hallander *et al.* (2010), dominance effects were also modeled and in principle can be included in our approach similar to GRM.

We hope that our work will facilitate exploration of other practical issues of Bayesian linear mixed models with polygenic effects, some of which are highlighted here.

6.1. Non-genetic effects

Although we have focused on the polygenic effects, their non-genetic counterparts can be an indispensable part of the research. For instance, BMI may be linked to lifestyle and

psychosocial factors such as diet, physical activity and mental health. SNP effects are now commonly derived as part of a GWAS from the so-called mixed linear effects model involving polygenic effects and SNP dosage as fixed effects. Gene-environment interactions are also important.

For non-genetic effects, g-prior (Zellner 1986) is often used. In our notation, this amounts to $\beta \sim MVN(\beta_0, a\sigma^2(X^\top X)^{-1})$ where β_0 is a hyperparameter and a a positive scalar often chosen to be the sample size, noting the use of a instead of g as in the literature is simply to avoid confusion with the polygenic effects g throughout this paper and elsewhere. The prior can facilitate model comparison since in the case of multiple linear regression closed form regression coefficients can be obtained but some undesirable property in model comparison has also been documented (e.g., Pericchi 2005).

6.2. Efficient implementation

The polygenic modeling would benefit greatly from a truly efficient Bayesian computation software system involving fine-tuned algorithms. Our limited experience showed that **JAGS** and **Stan** are feasible for moderate sample size ($N \approx 1,000$) but become very time-consuming when it gets larger. Besides approaches described in Section 4.1, **JAGS** can be compiled to use multicore facility. Recent versions of **rstan** actually have an option `cores` to automatically use all available cores. We do not attempt to elaborate this here as it is an active and evolving area with work such as Kruschke (2015) giving further information.

Our work suggests that a combination of generic Bayesian analysis systems such as **JAGS** and **Stan** together with specific software such as **BLR** will still be appealing. We also experimented with **MCMCglmm** and the function `MCMCgrm` both took considerably longer than **BLR**. Ahlinder and Sillanpaa (2013) made further attempt to speed up by treating β and u as nuisance parameters in the posterior distribution

$$\mathsf{P}(\beta, u, \sigma^2_\beta, \sigma^2_g, \sigma^2|y) \propto \mathsf{P}(y|\beta, u, \sigma^2)\mathsf{P}(\beta|\sigma^2_\beta)\mathsf{P}(u|\sigma^2_u)\mathsf{P}(\sigma^2_\beta)\mathsf{P}(\sigma^2_u)\mathsf{P}(\sigma^2)$$

so that $\mathsf{P}(\sigma^2_\beta, \sigma^2_u, \sigma^2|y) \propto \mathsf{P}(\sigma^2_\beta)\mathsf{P}(\sigma^2_u)\mathsf{P}(\sigma^2) \int \mathsf{P}(y|\beta, u, \sigma^2)\mathsf{P}(\beta|\sigma^2_\beta)\mathsf{P}(u|\sigma^2_u)d\beta du$ but the likelihood specification is still involved. Bayesian inference using Laplace approximation in the spirit of **INLA** is also available from **LaplacesDemon** (Statisticat, LLC. 2015) and a counterpart **LaplacesDemonCpp** (Statisticat and LLC. 2015) with an incremental inclusion of C++.

6.3. Missing outcome

It is more involved to allow for missing data. We did not address this explicitly and in general that is possible (**Stan** Development Team 2016c, p. 176). However, we took advantage of the built-in mechanism in **BLR**, For the Meyer data without filling the missing data, the results are as follows,

```
R> set.seed(1234567)
R> meyer <- within(meyer, {
+    yNa <- y
+    g1 <- ifelse(generation == 1, 1, 0)
+    g2 <- ifelse(generation == 2, 1, 0)
+    id <- animal
+    animal <- ifelse(!is.na(animal), animal, 0)
```

```
+    dam <- ifelse(!is.na(dam), dam, 0)
+    sire <- ifelse(!is.na(sire), sire, 0)
+  })
R> G <- kin.morgan(meyer)$kin.matrix * 2
R> library("regress")
R> r <- regress(y ~ -1 + g1 + g2, ~G, data = meyer)
R> r
R> library("BLR")
R> attach(meyer)
R> X <- as.matrix(meyer[c("g1","g2")])
R> m <- BLR(yNa, XF = X, GF = list(ID = 1:nrow(G), A = G),
+    prior = list(varE = list(df = 1, S = 0.25),
+    varU = list(df = 1, S = 0.63)),
+    nIter = 5000, burnIn = 500, thin = 1, saveAt = "meyer.BLR")
R> with(r, h2G(sigma, sigma.cov))
Vp = 104.091 SE = 9.925092
h2G = 0.3042677 SE = 0.1147779
R> names(m)
 [1] "y"       "weights" "mu"      "varE"    "yHat"    "SD.yHat"
 [7] "whichNa" "fit"     "bF"      "SD.bF"   "u"       "SD.u"
[13] "varU"    "prior"   "nIter"   "burnIn"  "thin"
R> attach(m)
R> yHat[whichNa]
numeric(0)
R> mu
[1] 327.9259
R> bF
        g1         g2
-105.11362  -89.52557
R> mu+bF
      g1       g2
222.8123 238.4004
R> varU
         [,1]
[1,] 29.66097
R> varE
[1] 74.08534
R> varU / (varU + varE)
         [,1]
[1,] 0.285899
```

with which we would be more comfortable. It seems that both frequentist and Bayesian approaches yielded smaller variance components compared to imputation of missing outcome

a priori. From the quantity `mu` and `bF` we are able to recover regression coefficients for the fixed effects comparable to what we have seen earlier. Furthermore, a vector `whichNa` indicates which observation has a missing outcome so that `yHat[whichNa]` contains predicted values for those missing outcomes.

GCTA can give heritability and standard error estimate for a quantitative trait based on a large number of SNPs. The documentation data involves a quantitative trait for 3,925 individuals and 1,000 SNPs, leading to $h^2(SE) = 0.022(0.009)$. We conducted a bootstrap experiment got an estimate of 0.191 (0.023), suggesting that some improvement can be made with respect to the usual likelihood estimate. Now we ran 5,000 burn-ins and 10,000 iterations with **BLR** and obtained 0.119 (0.001) and 95% HPD interval (0.098-0.142), still slightly higher than that based on REML.

Gaussian outcome is but one of many scenarios for which polygenic effects can be included. Our frequentist counterparts include **regress**, **pedigreemm** (Vazquez *et al.* 2010) and **coxme** (Therneau 2015), all in the R environment. They could involve problems with outcomes being binary, Poisson, time-to-event, etc. Our focus was on h^2 and there should be some similarity when we approach other indicators from the mixed models such as coefficient of determination (R^2) (Nakagawa and Schielzeth 2013).

Acknowledgments

The work derived from participation of the Genetic Analysis Workshops (GAW) involving porting the S-PLUS package **kinship**, developed at the Mayo Clinic, to R (Zhao 2005). We wish to thank Prof Terry Therneau and colleagues for many advices throughout the eight-year maintenance of **kinship**, whose functions are now contained in **bdsmatrix** (Therneau 2014), **coxme** and **kinship2** all available from `CRAN`. Two anonymous reviewers, associate editor Prof Donald Hedeker and the Editors have made numerous suggestions and recommendations leading to a much improved presentation.

References

Ahlinder J, Sillanpaa MJ (2013). "Rapid Bayesian Inference of Heritability in Animal Models without Convergence Problems." *Methods in Ecology and Evolution*, **4**(11), 1037–1046.

Anderson E, Bai Z, Bischof C, Blackford S, Demmel J, Dongarra J, Du Croz J, Greenbaum A, Hammarling S, McKenney A, Sorensen D (1999). ***LAPACK*** *Users' Guide*. 3rd edition. Society for Industrial and Applied Mathematics, Philadelphia, PA. ISBN 0-89871-447-8 (paperback).

Atkinson B, Therneau T (2012). ***kinship****: Mixed-Effects Cox Models, Sparse Matrices, and Modeling Data from Large Pedigrees*. R package version 1.1.4.

Bae HT, Perls TT, Sebastiani P (2014). "An Efficient Technique for Bayesian Modeling of Family Data Using the **BUGS** Software." *Frontiers in Genetics*, **5**, 390.

Burton PR, Scurrah KJ, Tobin MD, Palmer LJ (2005). "Covariance Components Models for Longitudinal Family Data." *International Journal of Epidemiology*, **34**(5), 1063–77; discussion 1077–9.

Clifford D, McCullagh P (2006). "The **regress** Function." *R News*, **6**(2), 6–10.

Damgaard LH (2007). "Technical Note: How to Use **WinBUGS** to Draw Inferences in Animal Models." *Journal of Animal Science*, **85**(6), 1363–1368.

de los Campos G, Perez P, Vazquez AI, Crossa J (2013). "Genome-Enabled Prediction Using the **BLR** (Bayesian Linear Regression) R Package)." In C Goodro, J van der Werf, B Hayes (eds.), *Genome-Wide Association Studies and Genomic Prediction*, chapter 12. Humana Press, New York.

Fisher RA (1918). "The Correlation between Relatives on the Supposition of Mendelian Inheritance." *Philosophical Transactions of the Royal Society of Edinburgh*, **52**, 399–433.

Gelman A (2006). "Prior Distributions for Variance Parameters in Hierarchical Models." *Bayesian Analysis*, **1**(3), 515–533.

Gelman A, Carlin JB, Stern HS, Dunson DB, Vehtari A, Rubin DB (2014). *Bayesian Data Analysis*. 3rd edition. Chapman & Hall/CRC.

Guo SW, Thompson EA (1991). "Monte-Carlo Estimation of Variance Component Models for Large Complex Pedigrees." *IMA Journal of Mathematics Applied in Medicine and Biology*, **8**(3), 171–189.

Hadfield JD (2010). "MCMC Methods for Multi-Response Generalized Linear Mixed Models: The **MCMCglmm** R Package." *Journal of Statistical Software*, **33**(2), 1–22.

Hallander J, Waldmann P, Wang C, Sillanpaa MJ (2010). "Bayesian Inference of Genetic Parameters Based on Conditional Decompositions of Multivariate Normal Distributions." *Genetics*, **185**(2), 645–54.

Holand AM, Steinsland I, Martino S, Jensen H (2013). "Animal Models and Integrated Nested Laplace Approximations." *G3.(Bethesda.)*, **3**(8), 1241–1251.

Intel (2013). *Intel Math Kernel Library*. Version 11.0.4.183, URL https://software.intel.com/en-us/intel-mkl.

Klimentidis YC, Vazquez AI, de Los CG, Allison DB, Dransfield MT, Thannickal VJ (2013). "Heritability of Pulmonary Function Estimated from Pedigree and Whole-Genome Markers." *Frontiers in Genetics*, **4**, 174.

Kruschke JK (2015). *Doing Bayesian Data Analysis: A Tutorial with R, **JAGS**, and **Stan***. 2nd edition. Elsevier Inc.

Lange K (2002). *Mathematical and Statistical Models for Genetic Data Analysis*. 2nd edition. Springer-Verlag, New York.

Lunn DJ, Thomas A, Best N, Spiegelhalter D (2000). "**WinBUGS** - A Bayesian Modelling Framework: Concepts, Structure, and Extensibility." *Statistics and Computing*, **10**(4), 325–337.

Meyer K (1989). "Restricted Maximum-Likelihood to Estimate Variance-Components for Animal-Models with Several Random Effects Using a Derivative-Free Algorithm." *Genetics Selection Evolution*, **21**(3), 317–340.

Morton NE, MacLean CJ (1974). "Analysis of Family Resemblance. 3. Complex Segregation of Quantitative Traits." *American Journal of Human Genetics*, **26**(4), 489–503.

Nakagawa S, Schielzeth H (2013). "A General and Simple Method for Obtaining R^2 from Generalized Linear Mixed-Effects Models." *Methods in Ecology and Evolution*, **4**(2), 133–142.

OpenBUGS Foundation (2015). ***OpenBUGS***. Cambridge. Version 3.2.3, URL http://www.openbugs.net/.

Perez P, de Los Campos G, Crossa J, Gianola D (2010). "Genomic-Enabled Prediction Based on Molecular Markers and Pedigree Using the Bayesian Linear Regression Package in R." *Plant Genome*, **3**(2), 106–116.

Pericchi LR (2005). "Model Selection and Hypothesis Testing based on Objective Probabilities and Bayes Factors." In CR Rao (ed.), *Handbook of Statistics*, volume 25, chapter 4, pp. 115–149. Elsevier Science Publishers, New York.

Plummer M (2013). ***JAGS***: *Just Another Gibbs Sampler*. Version 4-2.0, URL http://sourceforge.net/projects/mcmc-jags/.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. Vienna, Austria. R Foundation for Statistical Computing, URL http://www.R-project.org/.

Rue H, Martino S, Lindgren F, Simpson D, Riebler A, Krainski ET (2014). ***INLA***: *Functions which Allow to Perform Full Bayesian Analysis of Latent Gaussian Models Using Integrated Nested Laplace Approximaxion*. R package version 0.0-1404466487.

SAS Institute Inc (2014). *SAS/STAT Software*. Cary, NC. Version 9.3, URL http://www.sas.com/.

Sorensen D, Gianola D (2002). *Likelihood, Bayesian, and MCMC Methods in Quantitative Genetics*. Springer-Verlag, New York.

Stan Development Team (2016a). "**CmdStan**: the Command-Line Interface to **Stan**." Version 2.14.0, URL http://mc-stan.org/cmdstan.html.

Stan Development Team (2016b). ***RStan***: *the R Interface to* ***Stan***. R package Version 2.14.1, URL http://mc-stan.org/rstan.html.

Stan Development Team (2016c). ***Stan*** *Modeling Language*. Version 2.14.0, URL http://mc-stan.org/.

Statisticat, LLC (2015). ***LaplacesDemonCpp***: *C++ Extension for LaplacesDemon*. R package version 15.03.19, URL http://www.bayesian-inference.com/software.

Statisticat, LLC (2015). ***LaplacesDemon***: *Complete Environment for Bayesian Inference*. R package version 15.03.19, URL http://www.bayesian-inference.com/software.

Sturtz S, Ligges U, Gelman A (2005). "**R2WinBUGS**: A Package for Running **WinBUGS** from R." *Journal of Statistical Software*, **12**(3), 1–16. URL http://www.jstatsoft.org.

Su YS, Yajima M (2015). ***R2jags**: Using R to Run 'JAGS'*. R package version 0.5-7, URL http://CRAN.R-project.org/package=R2jags.

Tempelman RJ, Rosa GJM (2004). "Empirical Bayes Approaches to Mixed Model Inference in Quantitative Genetics." In AM Saxton (ed.), *Genetic Analysis of Complex Traits Using SAS*, chapter 7. SAS Institute Inc, Cary, NC.

Therneau T (2014). ***bdsmatrix**: Routines for Block Diagonal Symmetric matrices*. R package version 1.3-2, URL http://CRAN.R-project.org/package=bdsmatrix.

Therneau TM (2015). ***coxme**: Mixed Effects Cox Models*. R package version 2.2-4, URL http://CRAN.R-project.org/package=coxme.

Therneau TM, Sinnwell J (2015). ***kinship2**: Pedigree Functions*. R package version 1.6.4, URL http://CRAN.R-project.org/package=kinship2.

Varga RS (2004). *Gersgorin and His Circles*. Springer-Verlag.

Varona L, Vidal O, Quintanilla R, Gil M, Sanchez A, Folch JM, Hortos M, Rius MA, Amills M, Noguera JL (2005). "Bayesian Analysis of Quantitative Trait Loci for Boar Taint in a Landrace Outbred Population." *Journal of Animal Science*, **83**(2), 301–307.

Vazquez AI, Bates DM, Rosa GJM, Gianola D, Weigel KA (2010). "Technical Note: An R Package for Fitting Generalized Linear Mixed Models in Animal Breeding." *Journal of Animal Science*, **88**(2), 497–504.

Waldmann P (2009). "Easy and Flexible Bayesian Inference of Quantitative Genetic Parameters." *Evolution*, **63**(6), 1640–1643.

Waldmann P, Hallander J, Hoti F, Sillanpaa MJ (2008). "Efficient Markov Chain Monte Carlo Implementation of Bayesian Analysis of Additive and Dominance Genetic Variances in Noninbred Pedigrees." *Genetics*, **179**(2), 1101–12.

Yang J, Benyamin B, McEvoy BP, Gordon S, Henders AK, Nyholt DR, Madden PA, Heath AC, Martin NG, Montgomery GW, Goddard ME, Visscher PM (2010). "Common SNPs Explain a Large Proportion of the Heritability for Human Height." *Nature Genetics*, **42**(7), 565–569.

Yang J, Lee SH, Goddard ME, Visscher PM (2011). "**GCTA**: A Tool for Genome-Wide Complex Trait Analysis." *American Journal of Human Genetics*, **88**(1), 76–82.

Yi N, Xu S (2000). "Bayesian Mapping of Quantitative Trait Loci for Complex Binary Traits." *Genetics*, **155**(3), 1391–403.

Zellner A (1986). "On Assessing Prior Distributions and Bayesian Regression Analysis with g Prior Distributions." In P Goel, A Zellner (eds.), *Bayesian Inference and Decision Techniques: Essays in Honor of Bruno de Finetti*, volume 6, chapter 15, pp. 233–243. Elsevier Science Publishers, New York.

Zhao JH (2005). "Mixed-Effects Cox Models of Alcohol Dependence in Extended Families." *BMC Genetics*, **6 Suppl 1**, S127.

Zhao JH (2007). "**gap**: Genetic Analysis Package." *Journal of Statistical Software*, **23**(8), 1–18.

Zhao JH, Luan J (2012). "Mixed Modeling with Whole Genome Data." *Journal of Probability and Statistics*, **2012**, 1–16.

Affiliation:

Jing Hua Zhao, Jian'an Luan
MRC Epidemiology Unit
University of Cambridge School of Clinical Medicine
Box 285, Institute of Metabolic Science
Cambridge Biomedical Campus
Cambridge CB2 0QQ
United Kingdom
E-mail: jinghua.zhao@mrc-epid.cam.ac.uk, jianan.luan@mrc-epid.cam.ac.uk
URL: http://people.ds.cam.ac.uk/jhz22, http://www.mrc-epid.cam.ac.uk/people/

Peter Congdon
School of Geography and Life Sciences Institute
Queen Mary
University of London
Mile End Road
London E1 4NS
United Kingdom
E-mail: p.congdon@qmul.ac.uk
URL: http://www.geog.qmul.ac.uk/staff/congdonp.html

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd